Media Engineering
Game Engines

R. Weller

University of Bremen, Germany

cgvr.cs.uni-bremen.de

Mehr Argumente für Game Engineering

- Tools werden nicht nur in Spielen eingesetzt, sondern zunehmend auch in anderen Anwendungen
  - Serious Games
  - VR, AR
  - Mobile Apps
  - Kunst (Interaktive Installationen), Architektur
  - Forschung (z.B. Datenvisualisierung)

[Quelle: Ilumens, 2013]

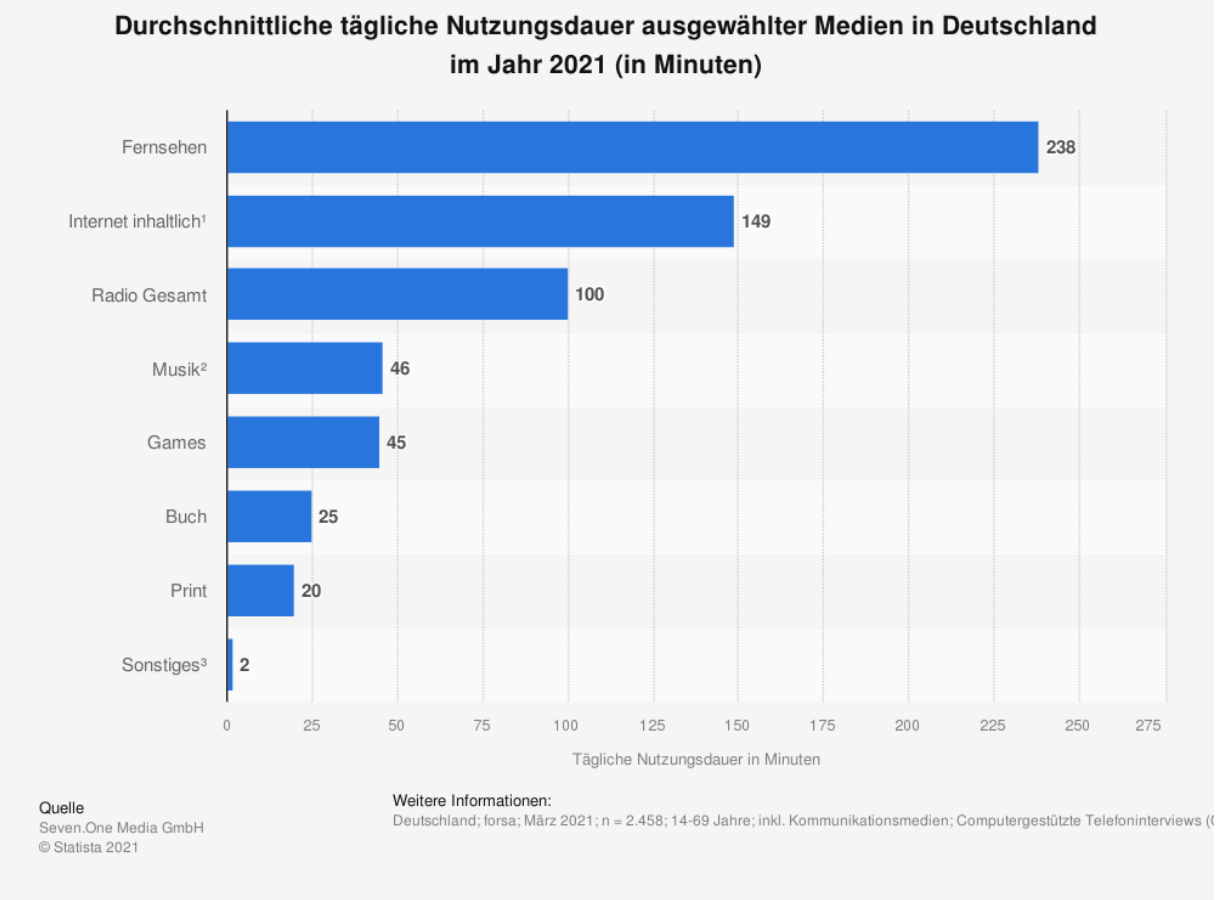
- Ähnliche Tools werden auch beim Film verwendet
  - Renderman (Pixar), Cinema 4D, 3ds Max
  - Blender hat sogar eine Game Engine eingebaut
  - Unterschied: Echtzeitfähigkeit (wobei Grafik-Engines für Filme oft einen reduzierte Echtzeitvorschau bieten)

Marktübersicht (Digitale) Medien in Deutschland

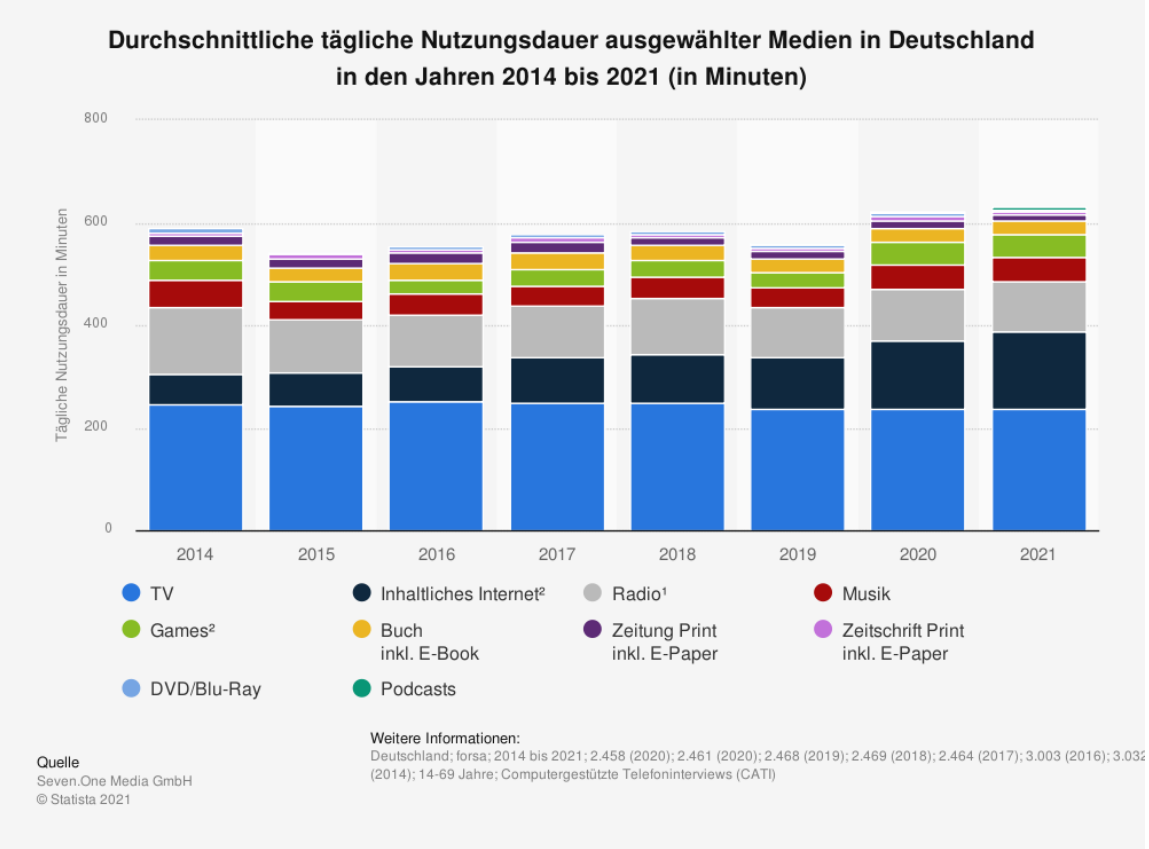
- Bücher
- Videospiele
- Fernsehen
- Zeitungen/Zeitschriften
- Hörfunk
- Internetvideo
- Musik

Gesamtumsatz 2019: ca. 56 Mrd

Mediennutzung in Deutschland 2021



Entwicklung Mediennutzung in Deutschland



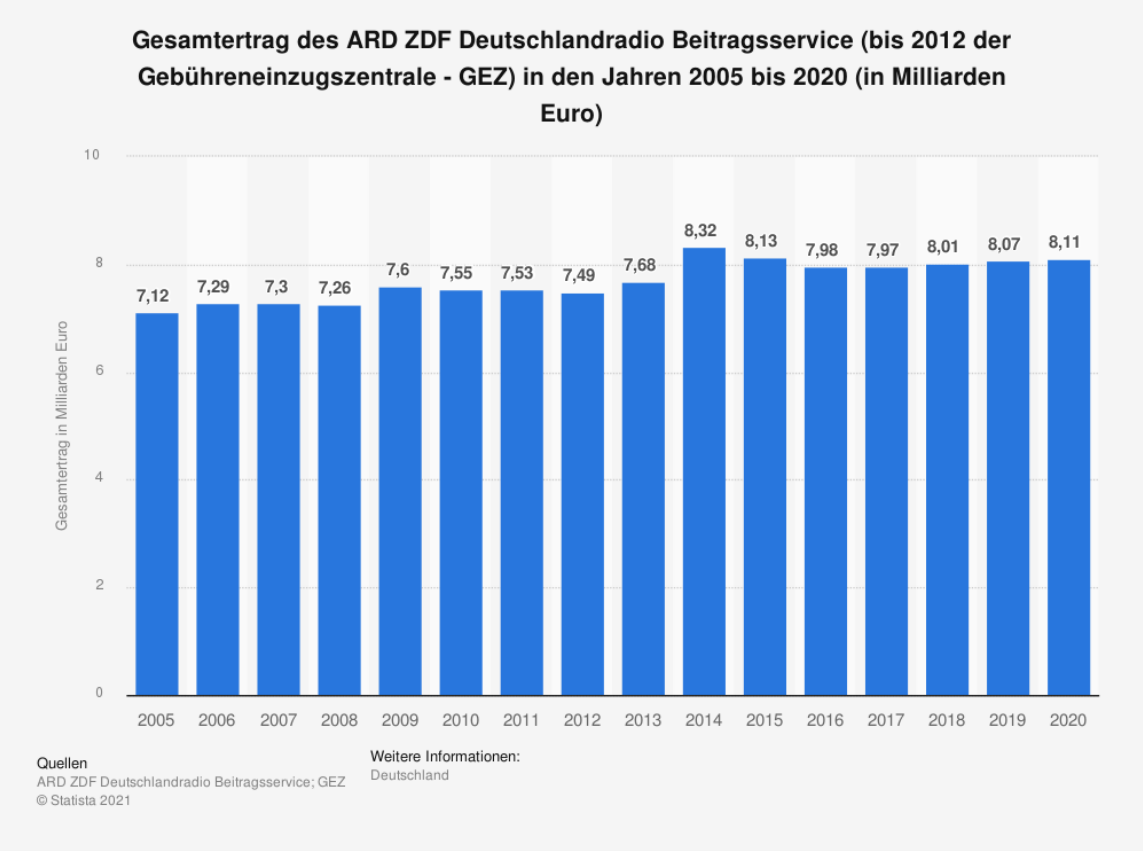
Marktübersicht (Digitale) Medien in Deutschland

Abb. 1 Gesamtumsätze der E&M-Branche in Deutschland (Seite 12)

Deutschland											Ø jährliches Wachstum
in Mio. Euro	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2020–25
Bücher	7.426	7.335	7.371	7.558	7.573	7.701	7.792	7.828	7.841	7.820	0,6 %
Veränderung (in %)		–1,23 %	0,49 %	2,54 %	0,20 %	1,69 %	1,18 %	0,47 %	0,16 %	–0,27 %	
Zeitungen & Zeitschriften	11.292	11.041	10.821	10.486	9.180	9.398	9.353	9.174	9.012	8.799	–0,8 %
Veränderung (in %)		–2,22 %	–1,99 %	–3,09 %	–12,46 %	2,37 %	–0,47 %	–1,92 %	–1,76 %	–2,36 %	
B2B	14.499	14.813	15.182	15.621	12.411	13.627	15.401	16.152	16.705	17.065	6,6 %
Veränderung (in %)		2,16 %	2,49 %	2,89 %	–20,55 %	9,79 %	13,02 %	4,88 %	3,42 %	2,15 %	
Musik, Radio, Podcast	4.557	4.646	4.540	4.696	3.181	3.942	4.913	5.196	5.355	5.488	11,5 %
Veränderung (in %)		1,95 %	–2,26 %	3,43 %	–32,27 %	23,92 %	24,63 %	5,78 %	3,05 %	2,49 %	
Kino	1.111	1.149	978	1.109	341	487	914	1.007	1.033	1.055	25,4 %
Veränderung (in %)		3,39 %	–14,90 %	13,43 %	–69,28 %	43,00 %	87,53 %	10,25 %	2,52 %	2,15 %	
Fernsehen	5.287	5.521	5.704	5.651	5.777	5.849	5.912	5.967	6.020	6.066	1,0 %
Veränderung (in %)		4,43 %	3,32 %	–0,94 %	2,24 %	1,24 %	1,07 %	0,94 %	0,88 %	0,77 %	
Internetvideo	763	866	1.047	1.319	1.749	1.906	2.086	2.249	2.378	2.506	7,5 %
Veränderung (in %)		13,49 %	20,87 %	25,96 %	32,63 %	9,00 %	9,43 %	7,78 %	5,74 %	5,40 %	
TV-Werbung	4.767	4.838	4.828	4.774	4.140	4.265	4.612	4.588	4.699	4.688	2,5 %
Veränderung (in %)		1,48 %	–0,21 %	–1,12 %	–13,28 %	3,02 %	8,15 %	–0,53 %	2,42 %	–0,22 %	
Onlinewerbung	6.167	6.970	7.732	8.532	8.918	9.499	10.214	10.998	11.778	12.451	6,9 %
Veränderung (in %)		13,02 %	10,94 %	10,34 %	4,53 %	6,51 %	7,54 %	7,67 %	7,09 %	5,72 %	
Videospiele und E-Sport	3.720	4.079	4.216	4.602	5.226	5.553	5.865	6.154	6.438	6.692	5,1 %
Veränderung (in %)		9,66 %	3,35 %	9,15 %	13,56 %	6,25 %	5,62 %	4,92 %	4,62 %	3,94 %	
Virtual Reality	16	32	47	63	81	98	118	143	174	211	21,2 %
Veränderung (in %)		95,73 %	47,25 %	33,27 %	27,44 %	21,82 %	20,68 %	20,69 %	21,50 %	21,51 %	
Außenwerbung	1.033	1.151	1.164	1.218	889	1.071	1.257	1.363	1.409	1.435	10,1 %
Veränderung (in %)		11,40 %	1,16 %	4,59 %	–27,03 %	20,52 %	17,39 %	8,44 %	3,33 %	1,84 %	
Gesamtumsatz (um Doppelzählungen bereinigt)	56.860	58.576	59.617	61.402	55.353	58.979	63.665	65.838	67.678	68.986	4,5 %
Veränderung (in %)		3,02 %	1,78 %	2,99 %	–9,85 %	6,55 %	7,95 %	3,41 %	2,79 %	1,93 %	

Quellen: PwC, Omdia.

TV-Gebühren in Deutschland



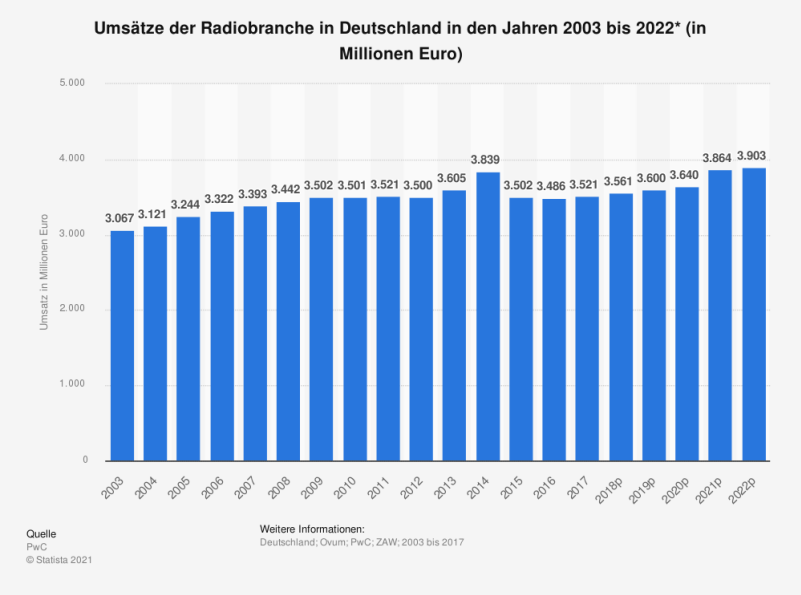
Radiomarkt

Kapitel Musik, Radio und Podcast

Abb. 15 Umsätze im gesamten Musik-, Radio- und Podcastmarkt (Seite 60)

in Mio. Euro	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025
Musikvertrieb inklusive Leistungsschutzrechte und Synchronisation	1.871	1.891	1.718	1.813	1.944	2.057	2.152	2.239	2.318	2.395
Livemusik	1.905	1.941	1.985	2.015	503	1.029	1.830	1.992	2.045	2.081
Radiowerbung	768	784	789	797	658	758	816	832	841	848
Podcastwerbung	13	28	48	71	76	97	116	134	151	165
Gesamt	4.557	4.646	4.540	4.696	3.181	3.942	4.913	5.196	5.355	5.488
Wachstumsrate in %		1,95	–2,26	3,43	–32,27	23,92	24,63	5,78	3,05	2,49

Quellen: PwC, Omdia, Bundesverband Musikindustrie.



Musikmarkt

Abb. 16 Umsätze im Musikmarkt (Seite 61)

in Mio. Euro	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025
Musikvertrieb	1.592	1.574	1.478	1.590	1.721	1.818	1.900	1.978	2.049	2.120
Livemusik	1.905	1.941	1.985	2.015	503	1.029	1.830	1.992	2.045	2.081
Leistungsschutzrechte	272	310	230	215	216	232	244	253	261	266
Synchronisation	7	7	10	8	7	7	8	8	8	9
Gesamt	3.776	3.833	3.703	3.827	2.447	3.086	3.981	4.231	4.363	4.476
Wachstumsrate in %		1,50	–3,40	3,36	–36,07	26,14	28,99	6,28	3,11	2,59

Quellen: PwC, Omdia, Bundesverband Musikindustrie.

Abb. 17 Umsätze des digitalen Musikvertriebes (Seite 62)

in Mio. Euro	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025
Streaming	385	549	705	909	1.133	1.298	1.436	1.562	1.673	1.777
Downloading	194	157	121	88	66	50	38	28	21	15
Klingeltöne/Ringbacks	24	21	18	15	12	10	8	6	5	4
Gesamt	603	726	844	1.013	1.211	1.358	1.482	1.596	1.698	1.796
Wachstumsrate in %		20,48	16,13	20,02	19,62	12,15	9,12	7,67	6,42	5,76

Quellen: PwC, Omdia, Bundesverband Musikindustrie.

Endergebnis Medienmarkt

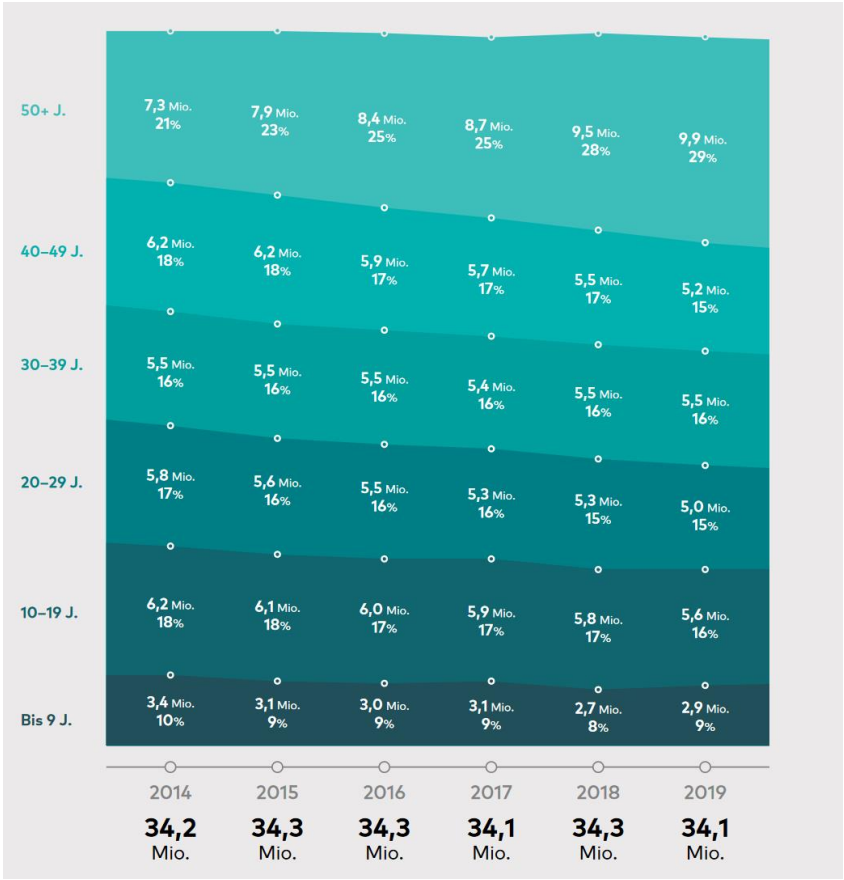
- Fernsehen (14,5M)
- Zeitungen/Zeitschriften (10,6M)
- Buch(7,6M)
- Videospiele(4,4M)
- Hörfunk(3,9M)
- Musik(3,8M)
- Internetvideo(1,1M)

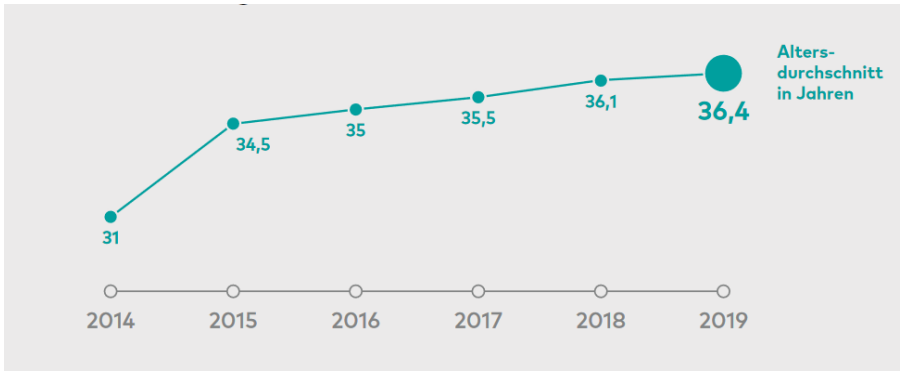
- Fernsehen(12,9M
- Buch(9,5M)
- Zeitungen(8,0M
- Zeitschriften(5,6
- Hörfunk(3,5M)
- Film(2,9M)
- Videospiele(1,9M
- Musik(1,5M)

Umsatzerlöse in den einzelnen Marktsegmenten

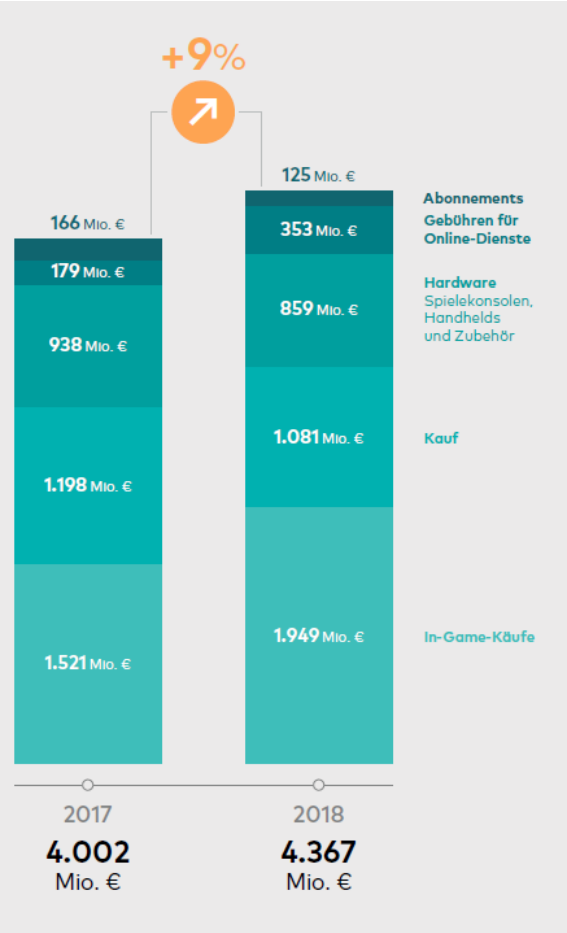
	2009	2010	2011	2012	2013	2014	2015
Film (in Mio. €)	2.707	2.659	2.746	2.827	2.861	2.891	2.966
Veränderung (in %)		–1,7	3,3	3,0	1,2	1,1	2,6
Fernsehen (in Mio. €)	11.225	11.733	12.023	12.511	12.949	13.456	13.595
Veränderung (in %)		4,5	2,5	4,1	3,5	3,9	1,0
Musik (in Mio. €)	1.575	1.489	1.483	1.435	1.452	1.454	1.459
Veränderung (in %)		–5,4	–0,4	–3,2	1,2	0,1	0,3
Hörfunk (in Mio. €)	3.502	3.501	3.507	3.497	3.588	3.711	3.660
Veränderung (in %)		0,0	0,2	–0,3	2,6	3,4	–1,4
Außenwerbung (in Mio. €)	811	843	897	868	891	917	942
Veränderung (in %)		3,9	6,4	–3,2	2,6	2,9	2,7
Onlinewerbung (in Mio. €)	3.332	3.769	4.249	4.670	5.126	5.541	5.932
Veränderung (in %)		13,1	12,7	9,9	9,8	8,1	7,1
Internetzugang (in Mio. €)	10.161	10.657	11.926	12.869	13.475	13.893	14.383
Veränderung (in %)		4,9	11,9	7,9	4,7	3,1	3,5
Zeitschriften (in Mio. €)	5.750	5.832	5.824	5.690	5.609	5.504	5.412
Veränderung (in %)		1,4	–0,1	–2,3	–1,4	–1,9	–1,7
Zeitungen (in Mio. €)	8.588	8.673	8.692	8.441	8.076	7.796	7.635
Veränderung (in %)		1,0	0,2	–2,9	–4,3	–3,5	–2,1
Bücher (in Mio. €)	9.695	9.734	9.601	9.523	9.540	9.547	9.583
Veränderung (in %)		0,4	–1,4	–0,8	0,2	0,1	0,4
Videospiele (in Mio. €)	1.895	2.013	2.096	1.967	1.932	2.066	2.157
Veränderung (in %)		6,2	4,1	–6,2	–1,8	6,9	4,4
Umsatzerlöse der gesamten Markt-segmente (in Mio. €)	**59.065**	**60.668**	**62.752**	**63.948**	**65.130**	**66.385**	**67.306**
Veränderung (in %)		2,7	3,4	1,9	1,8	1,9	1,4

Quellen: FFA, BVV, ZAW, OVK, GfK-Gruppe, Bundesverband Musikindustrie, VDZ, IVW, Fachpresse-S
deutschen Buchhandels, PwC, Ovum.

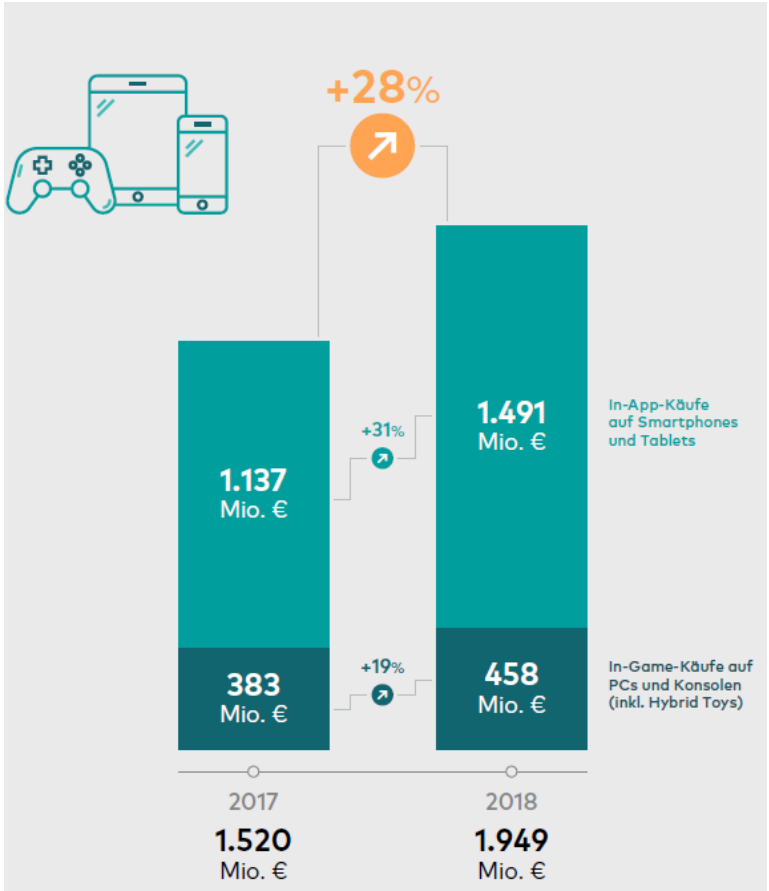

50+ J.
40–49 J.
30–39 J.
20–29 J.
10–19 J.
Bis 9 J.
7,3 Mio. 21%
7,9 Mio. 23%
8,4 Mio. 25%
8,7 Mio. 25%
9,5 Mio. 28%
9,9 Mio. 29%
6,2 Mio. 18%
6,2 Mio. 18%
5,9 Mio. 17%
5,7 Mio. 17%
5,5 Mio. 17%
5,2 Mio. 15%
5,5 Mio. 16%
5,5 Mio. 16%
5,5 Mio. 16%
5,4 Mio. 16%
5,5 Mio. 16%
5,5 Mio. 16%
5,8 Mio. 17%
5,6 Mio. 16%
5,5 Mio. 16%
5,3 Mio. 16%
5,3 Mio. 15%
5,0 Mio. 15%
6,2 Mio. 18%
6,1 Mio. 18%
6,0 Mio. 17%
5,9 Mio. 17%
5,8 Mio. 17%
5,6 Mio. 16%
3,4 Mio. 10%
3,1 Mio. 9%
3,0 Mio. 9%
3,1 Mio. 9%
2,7 Mio. 8%
2,9 Mio. 9%
2014
2015
2016
2017
2018
2019
34,2 Mio.
34,3 Mio.
34,3 Mio.
34,1 Mio.
34,3 Mio.
34,1 Mio.


Alters-durchschnitt in Jahren
31
34,5
35
35,5
36,1
36,4
2014
2015
2016
2017
2018
2019

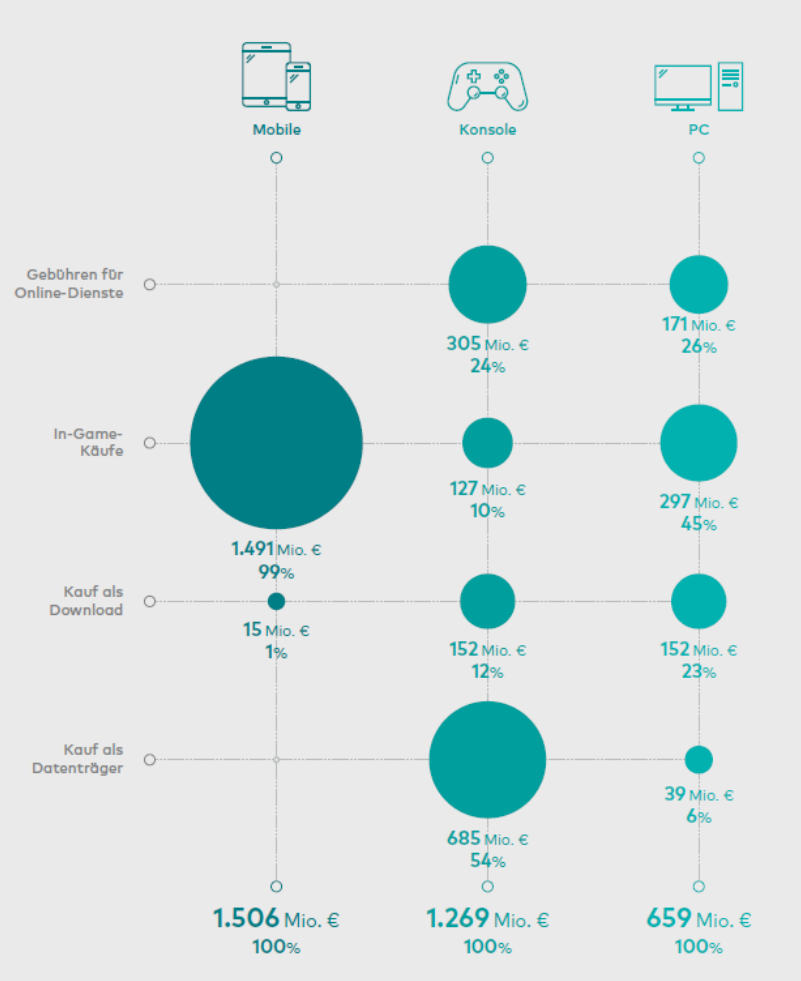
Faktencheck



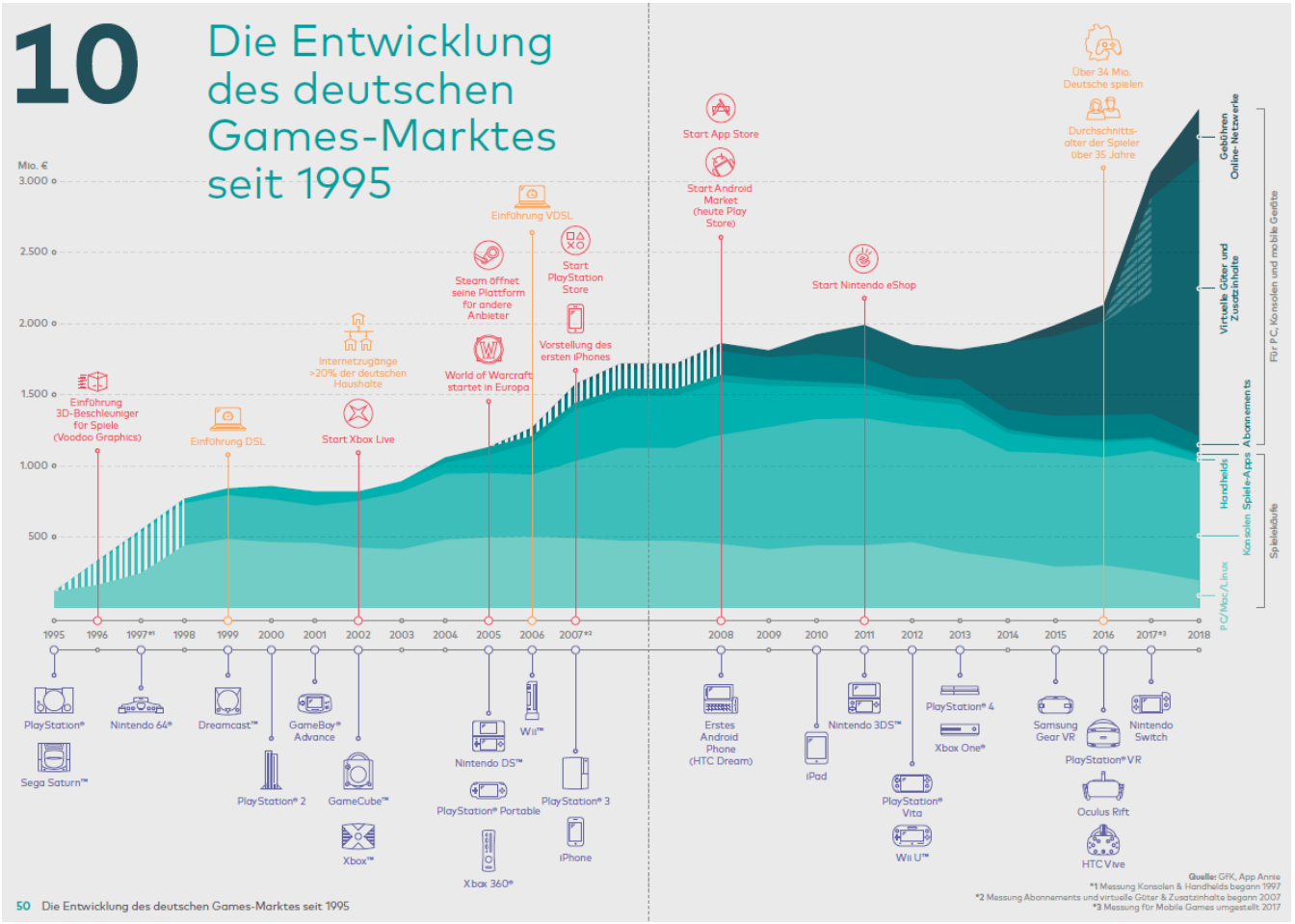
Faktencheck



Faktencheck

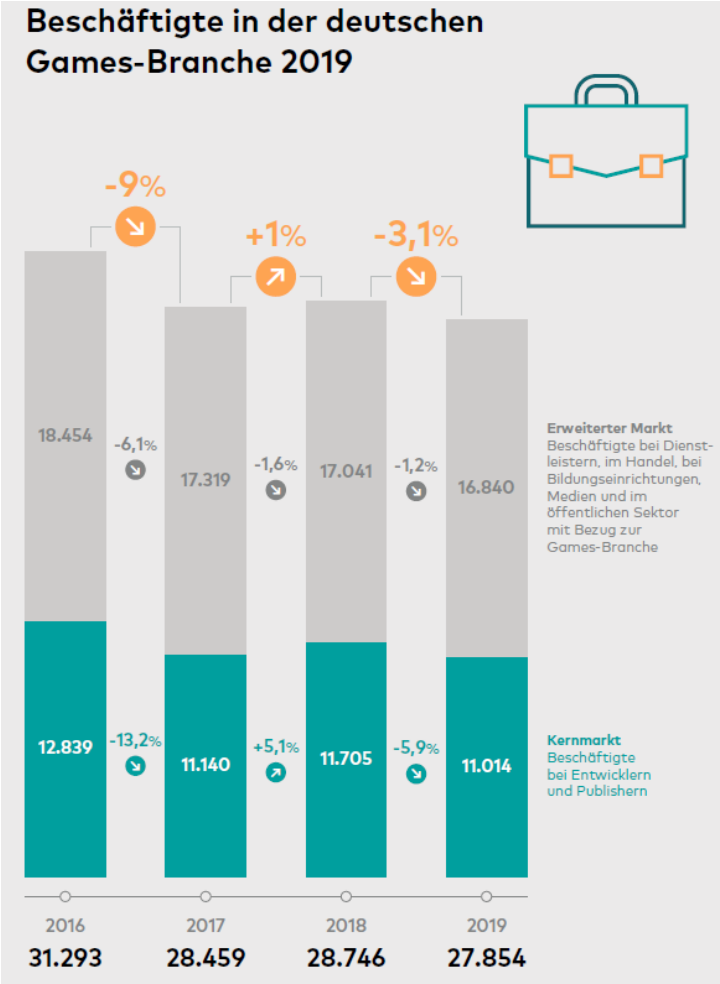


Faktencheck



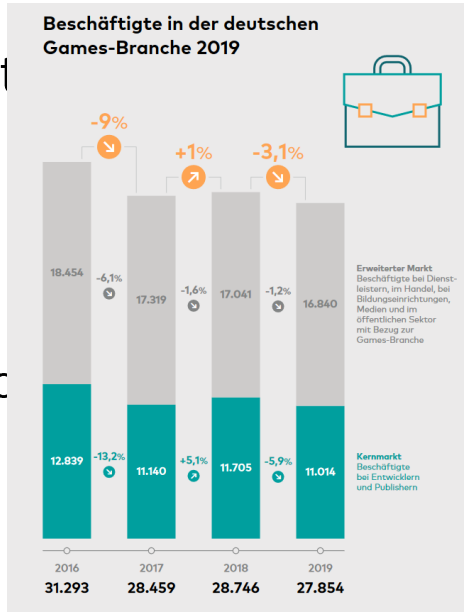
Faktencheck

Faktencheck

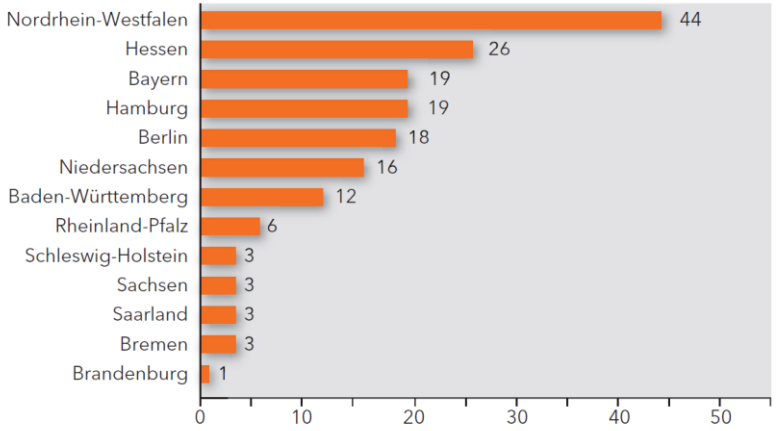


Videospielstandort Deutschland

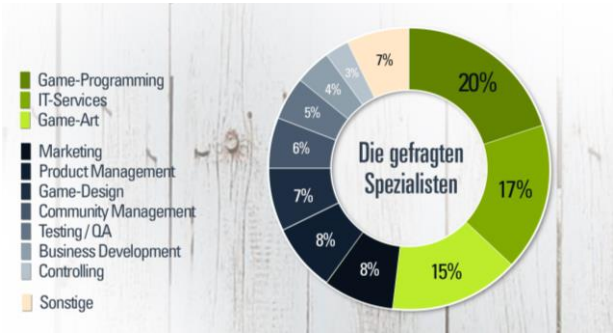
- 686 Unternehmen, ca 10000 Beschäftigt
  - Kerngeschäft Entwickler und Publisher:
    275 Betriebe (mit > 5 Angestellten)
    6000 Beschäftigte
    > 1500 offene Stellen (auf games-career.cc



Entwicklerstudios für Videospiele in Deutschland

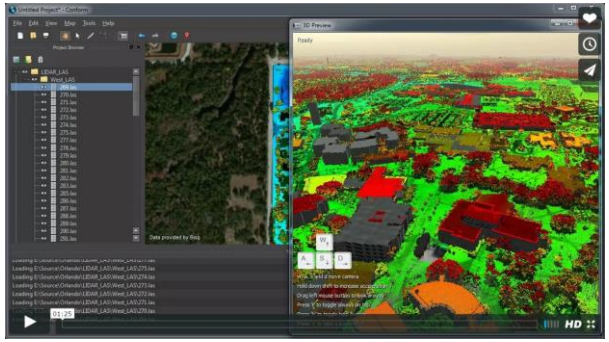


[Quelle: mediabiz.de]



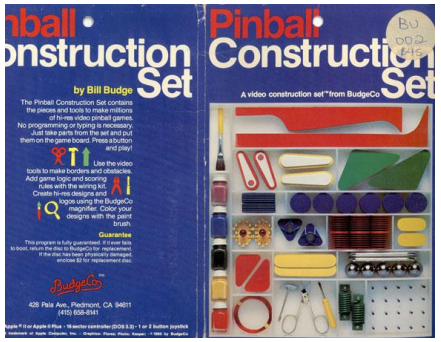
[Quelle: games-career.com]

Mehr Argumente für Game Engineering



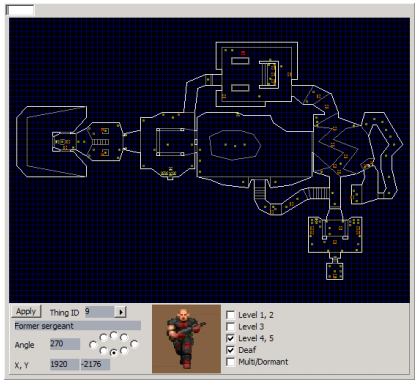
Spieleentwicklung - Prä-Historisches

- Ab ca 1977: Arcade-Spiele
  - Hardware sehr langsam
    - Spezialhardware
  - Zeitaufwändige Anpassung an Zielmaschine
    => Kaum Wiederverwendung von Code möglich
- Ab ca 1983: Game Creation Systeme
  - Sehr spezielle Tools für bestimmte Genres
  - Meist für kleinere Indie-Titel
  - Beispiele: Pinball Construction KIT (1983), Adventure Construction Set (1984), Wargame Construction Set (1986), Shoot'Em-Up Construction Kit (1987), RPG Maker (1988)



Historisches – Der Beginn

- Der Begriff „Game Engine“ wurde ab Mitte der 1990er im Zusammenhang mit 3D-FPS (First-Person-Shootern) wie Wolfenstein 3D und Doom verwendet.
  - Trennung von Technik und eigentlichem „Game Content“ wie Texturen und Leveln
  - Doom bot den Benutzern die Möglichkeit, eigene Modifikationen (z.B. Level, Waffen, Gegner) des Contents zu entwerfen und miteinander zu teilen
  - Kommerzielle Anwender konnten den Programmcode lizensieren und damit erstellte Spiele auch verkaufen
    - Doom II (1994), Heretic (1996), Hexen (1995), Strife (1996), Check Quest (1996)
    - HacX: Twitch n Kill (1997)
    - Cruis‘n Velocity (2001)
    - Dark Arena (2002)

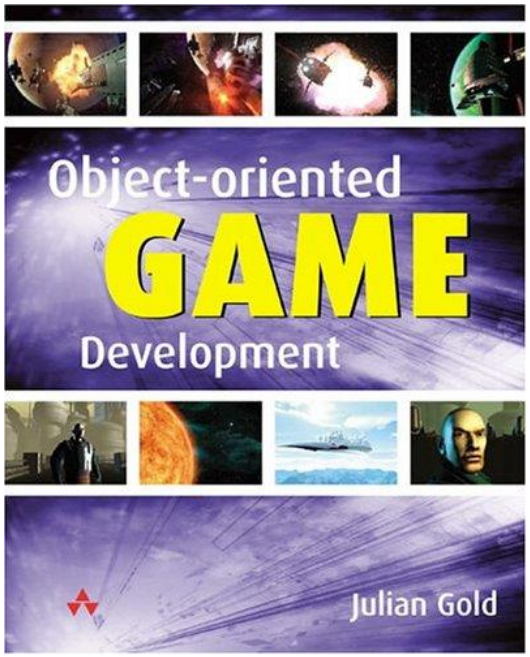


Historisches – Die Entwicklung bis heute

- Der Erfolg von Doom und Quake wurde von vielen Spieleherstellern kopiert (z.B. Beigabe von Leveleditoren für Nutzercontent)
  - Z.B.: Duke Nukem 3D, Warcraft II
- Spätere Game Engines wurden extra für die Lizenzierung entwickelt (aber immer noch von Firmen, die hauptsächlich durch den Verkauf ihrer eigenen Spiele Geld verdienten)
  - Z.B. Quake II (1997) und Unreal (1998)
- Später wurden spezielle Game Engines nur für den Verkauf entwickelt und die Spiele waren nur noch Demo für die Engine
  - CryEngine (2004)
  - Oder die Engineentwickler stellten gar keine Spiele mehr her (Vollständige Trennung von Game Engine und Game Content)
    - Unity (seit 2005)

Definition: Game Engine

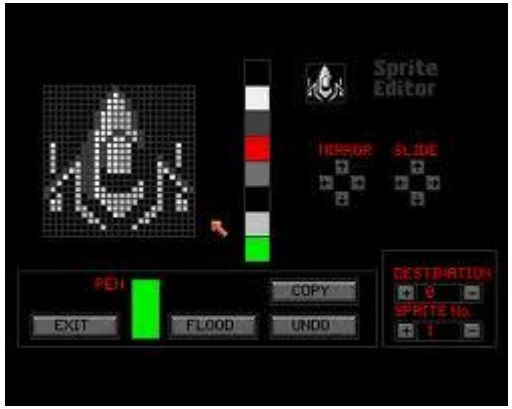
"A series of modules and interfaces that allows a development team to focus on produce *game-play content*, rather than *technical content*." - **Julian Gold, Object-Oriented Game Dev.**

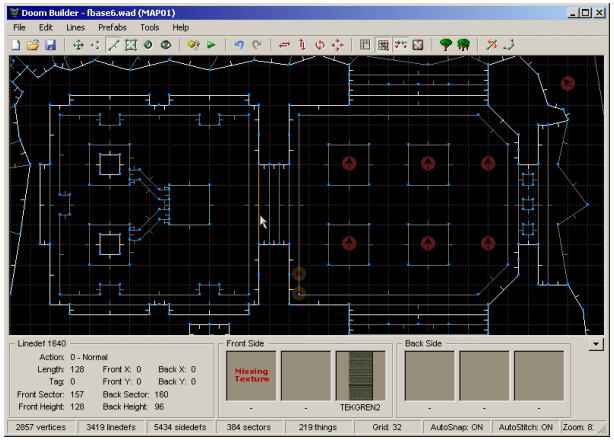


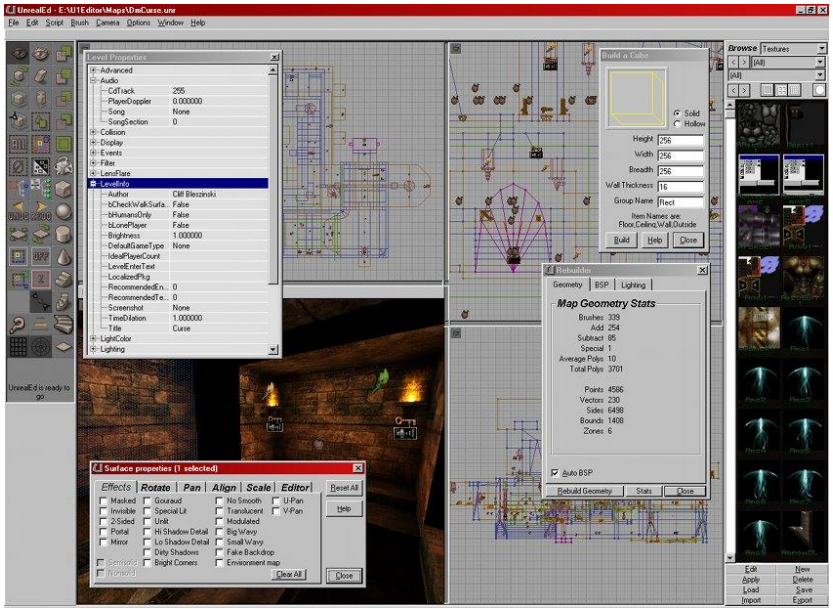
Warum verwendet man Game Engines?

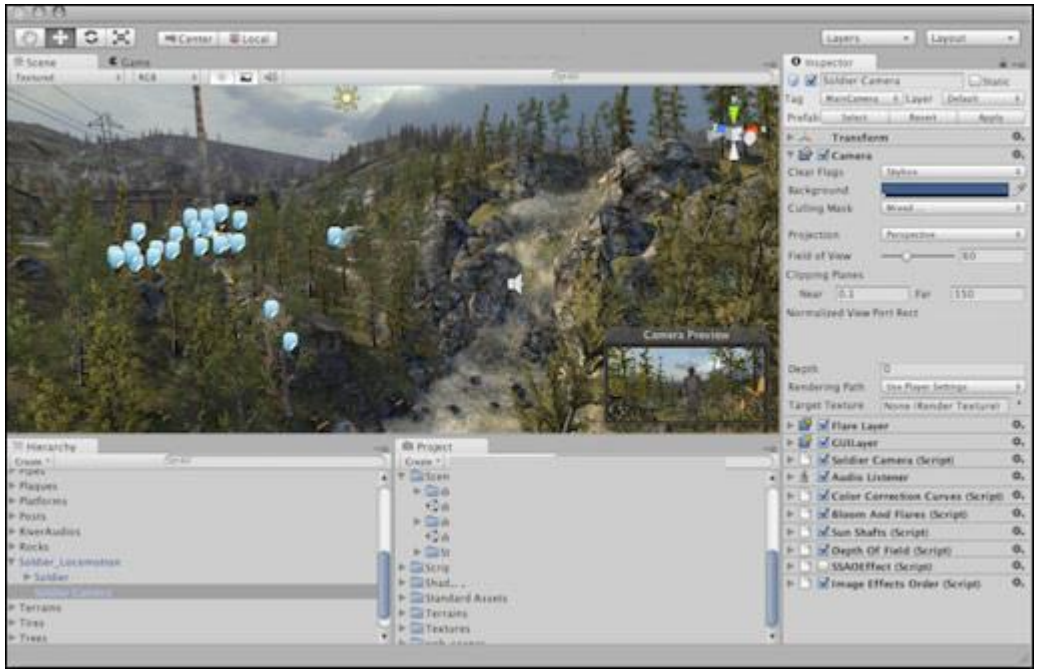
- Bieten einen einfachen Ansatz die Entwicklung des Spieleinhalts von den darunter liegenden technischen Aspekten zu abstrahieren
- Erleichtert die Wiederverwendbarkeit von Code
- Ermöglicht die gleichzeitige Entwicklung für mehrere Zielplattformen
  - PC/Mac
  - Konsolen
  - Mobile Geräte
    - Android/iOS

Game Engines im Wandel der Zeit









Game Engines im Wandel der Zeit – Komplexität (?)

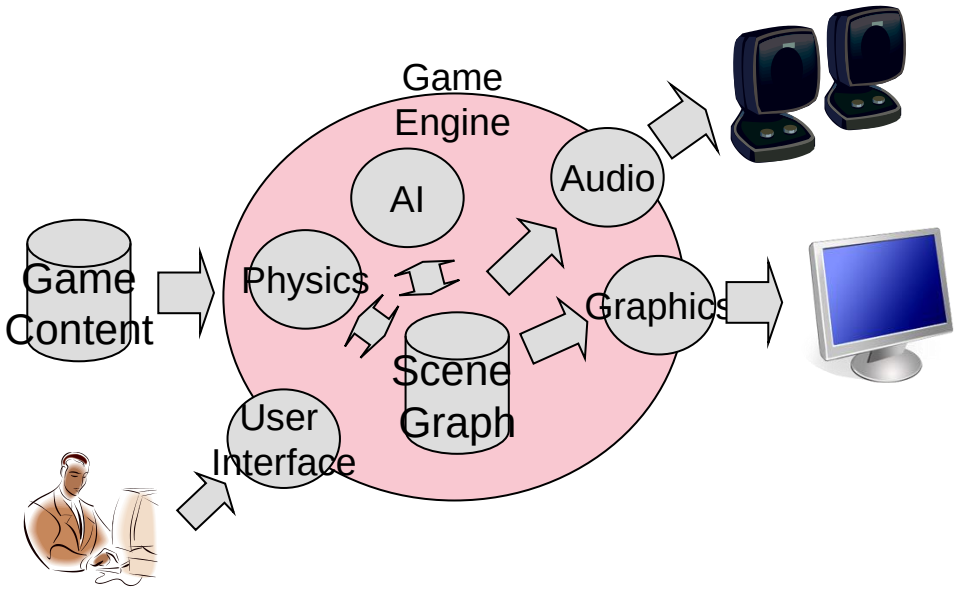
id Tech 1 (1999)	79k
id Tech 2 (2001)	138k
id Tech 3 (2005)	329k
id Tech 4 (2011)	586k
UE4 v4.6 (2015)	1964k

- Used cloc
- Only counting C, C++ and header files.

"Measuring software productivity by lines of code is like measuring progress on an airplane by how much it weighs." - Bill Gates

Was ist eine Game Engine?

- Softwaretechnologie zum Erstellen von Computerspielen
- Behinhaltet unter anderem zahlreiche Softwaremodule die für die meisten Spiele benötigt werden:
  - Graphik
  - Physik
  - Sound
  - Scripting
  - Animation
  - Künstliche Intelligenz (KI)
  - Netzwerkunterstützung
  - Benutzerschnittstellen



Die Module im Überblick: Rendering Engine

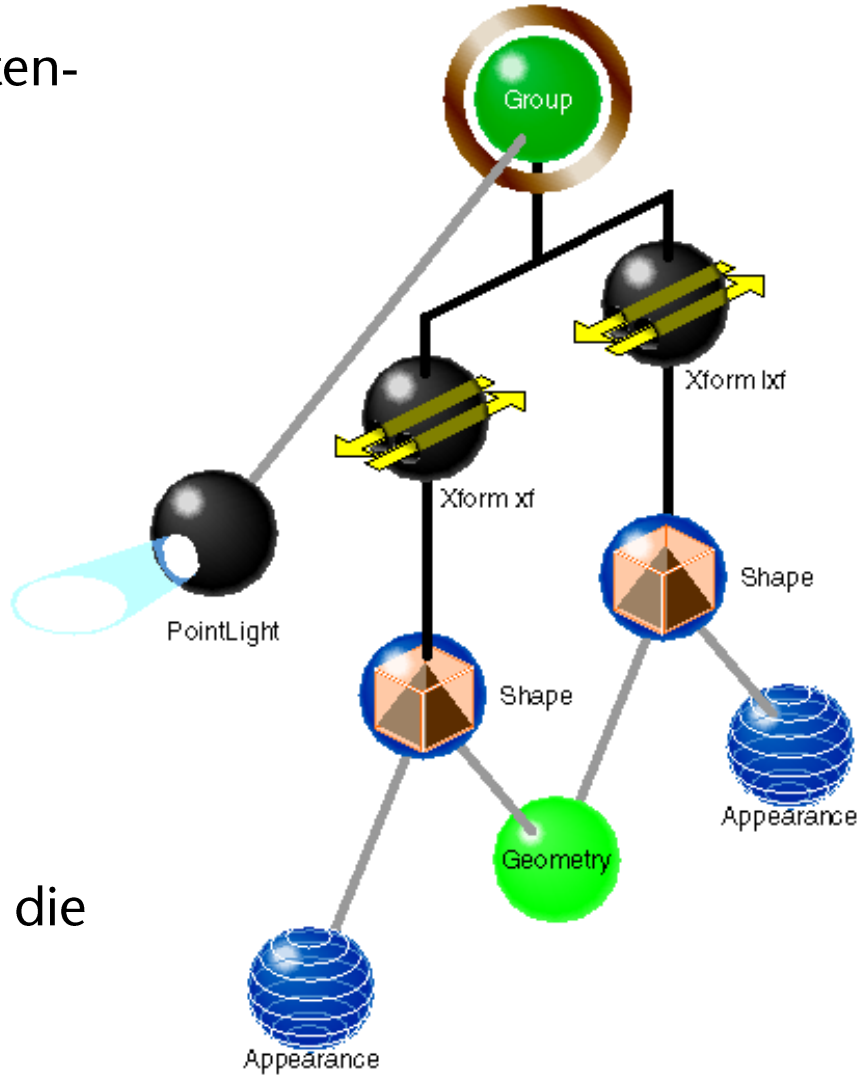
- Aufgabe der Rendering Engine ist es, die vom Benutzer erstellten 3D-Objekte hübsch darzustellen
- Sie ist meist das größte und komplexeste Modul einer Game Engine
- Oft wird sie in mehrere Untermodule unterteilt:
  - Renderer
  - Szenengraph
  - Visuelle Effekte
  - GUI

Rendering Engine: Renderer

- Die Aufgabe des Renderers ist es, eine Menge von geometrischen Primitiven so schnell wie möglich darzustellen
  - Geometrische Primitiven sind beispielsweise: 3D Dreiecksnetze, Linien, Punkte, Partikel
- Der Renderer stellt darüber hinaus ein Basissystem für die Beleuchtung und die Materialverwaltung bereit
- Dabei greift er meist direkt auf die Grafikhardware zu oder er verwendet Plattform-unabhängige Zwischenschichten wie OpenGL oder DirectX

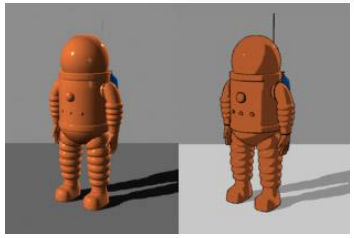
Rendering Engine: Szenengraph

- Üblicherweise eine baumartige Datenstruktur um die Objekte einer Spielszene für das Rendering (und andere Spielaspekte) zu verwalten
  - Objekte
    - Transformationen
    - Materialien
  - Lichtquellen
  - Kameras
- Unterstützt meist Culling (d.h. Das Abschneiden von Teilen der Szene, die nicht im Sichtbereich)



Rendering Engine: Visuelle Effekte

- Spezialeffekte, die nicht direkt vom Renderer (bzw der Zwischenschicht OpenGL oder DirectX) unterstützt werden
- Oft durch mitgelieferte Shader-Programme implementiert
- Beispiele:
  - Partikeleffekte (Feuer, Rauch, Explosionen)
  - Dynamische Schatten
  - Non-Photorealistic-Rendering (z.B. Cel Shading)
  - Environmental Mapping (Spiegelungen der Umgebung)
  - Post-Render-Effekte
    - HDR, FSAA, Farbkorrektur

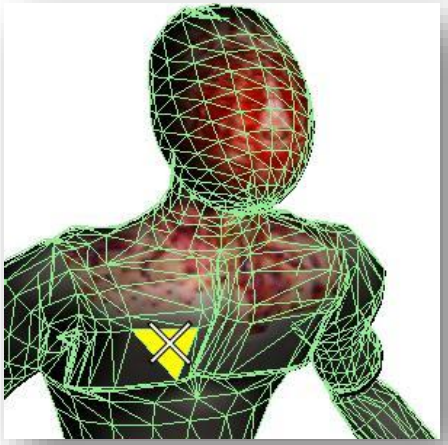


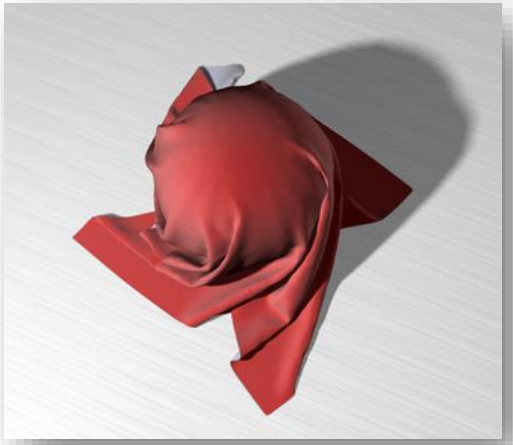
Rendering Engine: GUI

- Darstellung von 2D-Elementen
  - Z.B: Menus
  - Heads-Up-Display (HUD)
  - In-Game 2D-Benutzerinterfaces (z.B. Inventar)
  - Darstellung von Debugging-Informationen im Spiel

Physik

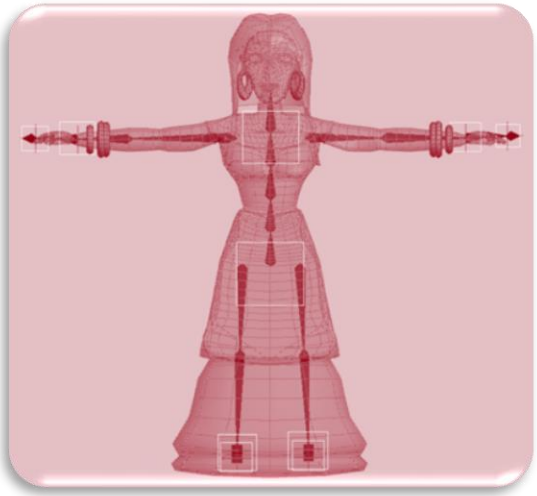
- Sorgt dafür, dass sich Objekte in einer Szene gemäss den Newton'schen Gesetzen der Physik verhalten (also, z.B., dass einem der Apfel auf den Kopf fällt)
- Definiert Masse, Kräfte und Geschwindigkeiten für die Objekte
- Kollisionsdetektion erkennt, wann Objekte zusammenstossen
- Verwendet oft vereinfachte Modelle der Physik um Echtzeitfähigkeit zu garantieren





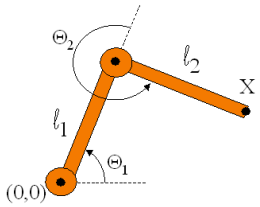
Animation

- Animation der Spielcharaktere
  - Meist Skelettanimation
  - Oft werden Daten unterstützt, die per Motion-Capturing gewonnen werden
- Überblenden einzelner Posen (Keyframes)
- Übertragen der Bewegung auf das darüberliegende Dreiecksnetz (Skinning)
- Teilweise auch inverse Kinematiken oder Forward-Kinematiken



Inverse Kinematics

- Animator specifies end-effector positions: X
- Computer finds joint angles: Θ_1 and Θ_2:



$$\Theta_2 = \cos^{-1}\left(\frac{x^2 + x^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right)$$

$$\Theta_1 = \frac{-(l_2 \sin(\Theta_2))x + (l_1 + l_2 \cos(\Theta_2))y}{(l_2 \sin(\Theta_2))y + (l_1 + l_2 \cos(\Theta_2))x}$$

Sound

- Abspielen von Sounds
- Hintergrundmusik (evtl szeneabhängiges Überblenden)
- Spielgeräusche (Motorengeräusche, Reifenquietschen, Waffenklirren,Schreie, Sprache,...)
  - Triggerbasiert
  - Abhängig von Spielphysik (Kollisionen)
- Spielgeräusche werden im 3D-Raum positioniert und mit Effekten versehen (Hall, Delay,...)
- Entscheidend ist hier oft die Anzahl

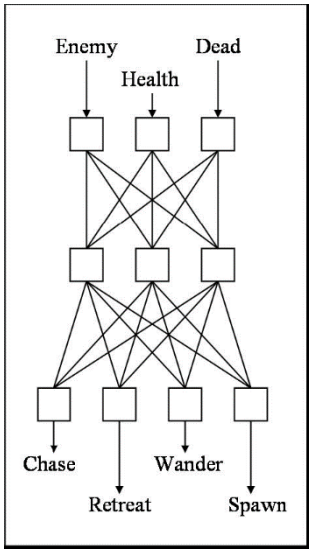
  und Art der Unterstützten Formate

  und Effekte (MP3, WAV, ogg,...)

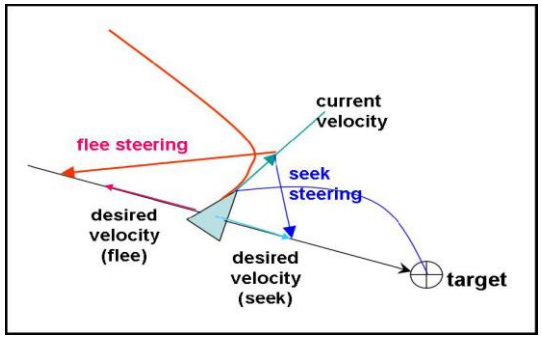
Künstliche Intelligenz (KI)

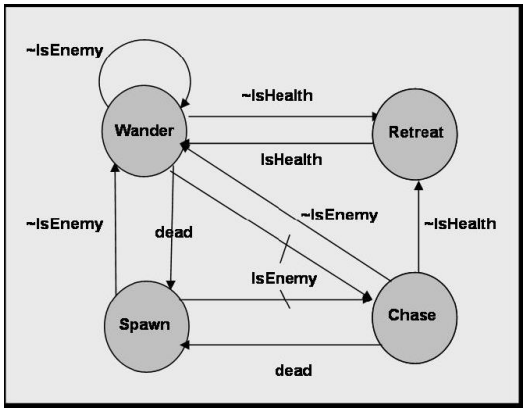
- Sorgt dafür, dass sich NPCs (aber auch Spieler) glaubhaft in der Spielewelt bewegen und verhalten
- Bessere Engines bieten Algorithmen aus der KI
  - Endliche Zustandsautomaten
  - Neuronale Netzwerke (lernen aus dem Verhalten des Spielers)
- Gruppenverhalten
  - Schwarmverhalten
  - Fluchtverhalten
- Wegfindung

Neural Network

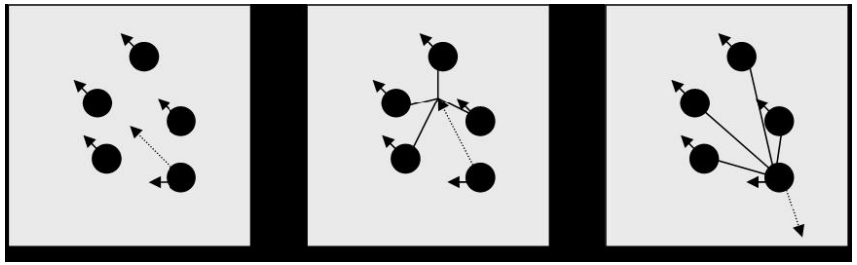


Seek and Flee Behaviour





Finite State Automata



Flocking

Scripting

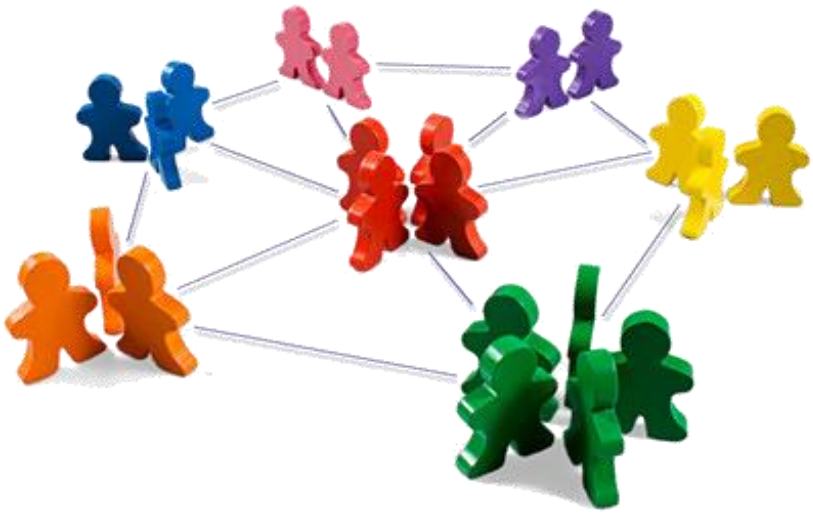
- Meist einfache, interpretierte Skriptsprache die einfachen Zugriff auf Teile (oder die gesamte) Engine erlaubt (dadurch muss das Spiel nicht bei jeder Änderung neu kompiliert werden)
- Z.B. Veränderung der Spielmechaniken durch Definition von
  - Events
  - Auszeichnungen
  - KI-Erweiterungen
- Kann eine Engine-unabhängige Sprache sein oder auch eine selbstentwickelte (Unreal-Scripting Language)

Benutzerschnittstellen

- Spiele unterstützen oft verschiedene Eingabegeräte
  - Keyboard und Maus
  - Gamepad
  - Spezialcontroller (Kinect, Lenkräder, Flighsticks, billige Plastikgitarrenimitate mit vier bunten Knöpfen, Wii Fit Board,...)
- Dieses Modul abstrahiert das Mapping von logischen Spielfunktionen und dem physikalischen Controller

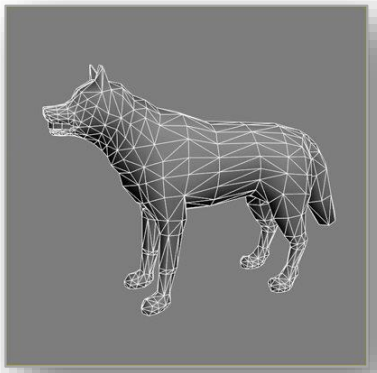
Netzwerkfunktionen

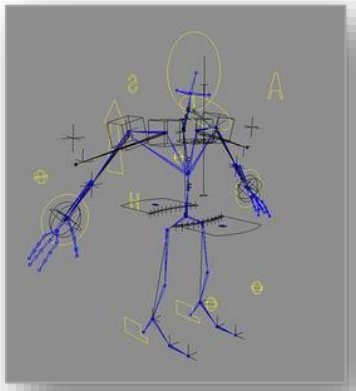
- Unterstützung von Multiplayer-Spielen, wobei meist nur direkte Kommunikation zwischen den Spielern unterstützt wird
  - Dies ist zu unterscheiden von Massively Multiplayer Games (MMOGs), bei denen tausende Spieler gleichzeitig in einer gemeinsamen Welt interagieren. Dazu werden meist riesige Serverfarmen benötigt
- Stellt einen gemeinsamen, eindeutigen Zustand auf allen beteiligten Rechnern sicher
  - Dazu werden meist ausgefeilte Caching-Strategien verwendet

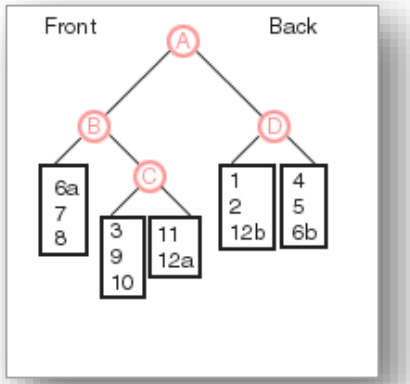


Resourcen-Manager

- Game Engine-eigenes Tool zum verwalten des Contents
  - 3D Objekte
  - Texturen
  - Fonts
  - Skeletanimationen
  - Levelbeschreibungen (Maps)
  - Sounds



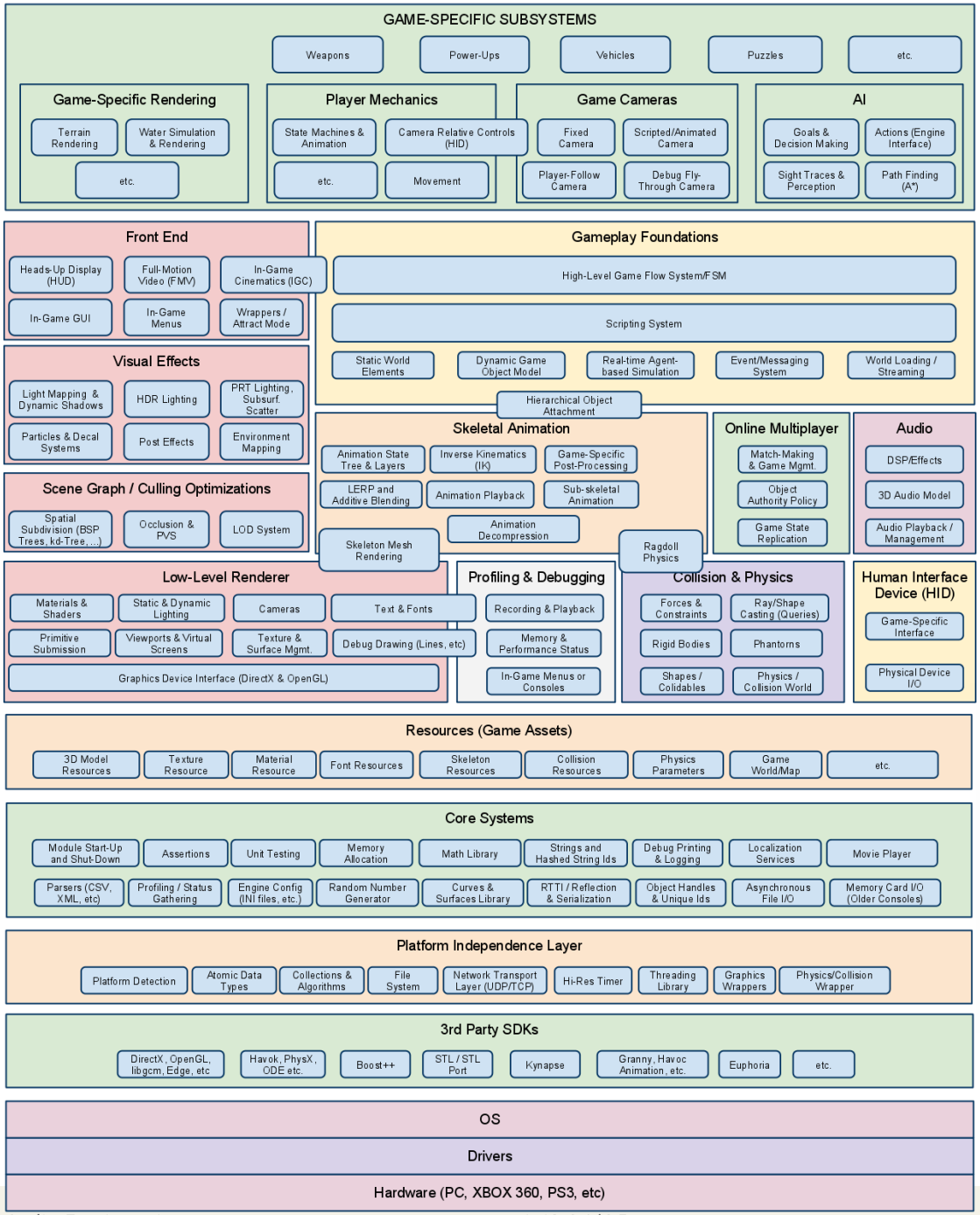


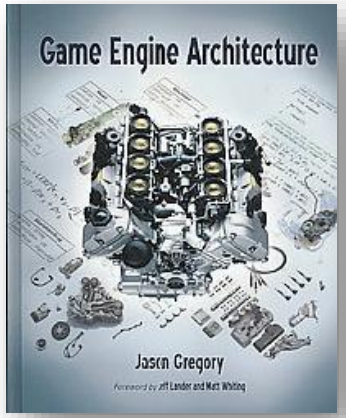


Core System

- Meist sehr hardwarenahe Utilities und Support-Klassen
  - Mathematik-Library
  - Zufallszahlengenerator
  - Funktionen zum Abspielen von Filmen
  - Parser
  - File I/O
  - Tools für die Übersetzung in andere Sprachen
  - Debugging/Profiling-Tools
  - Multithreading
  - Speicherverwaltung
  - Assertions
  - ...

Game Engine Architecture Blocks (complete?)





Game Engine Architecture, by Jason Gregory, 2009, AK Peters, ISBN: 1-5688-1413-5.

Die Game-Loop

- While user input not exit
  - Update Scene-Graph via User-Input
  - Update Scene-Graph with network cache
  - Update Scene-Graph via KI
  - Update Scene-Graph via Physics and Animation
  - Render Scene-Graph to Screen via Graphics System
- Endwhile

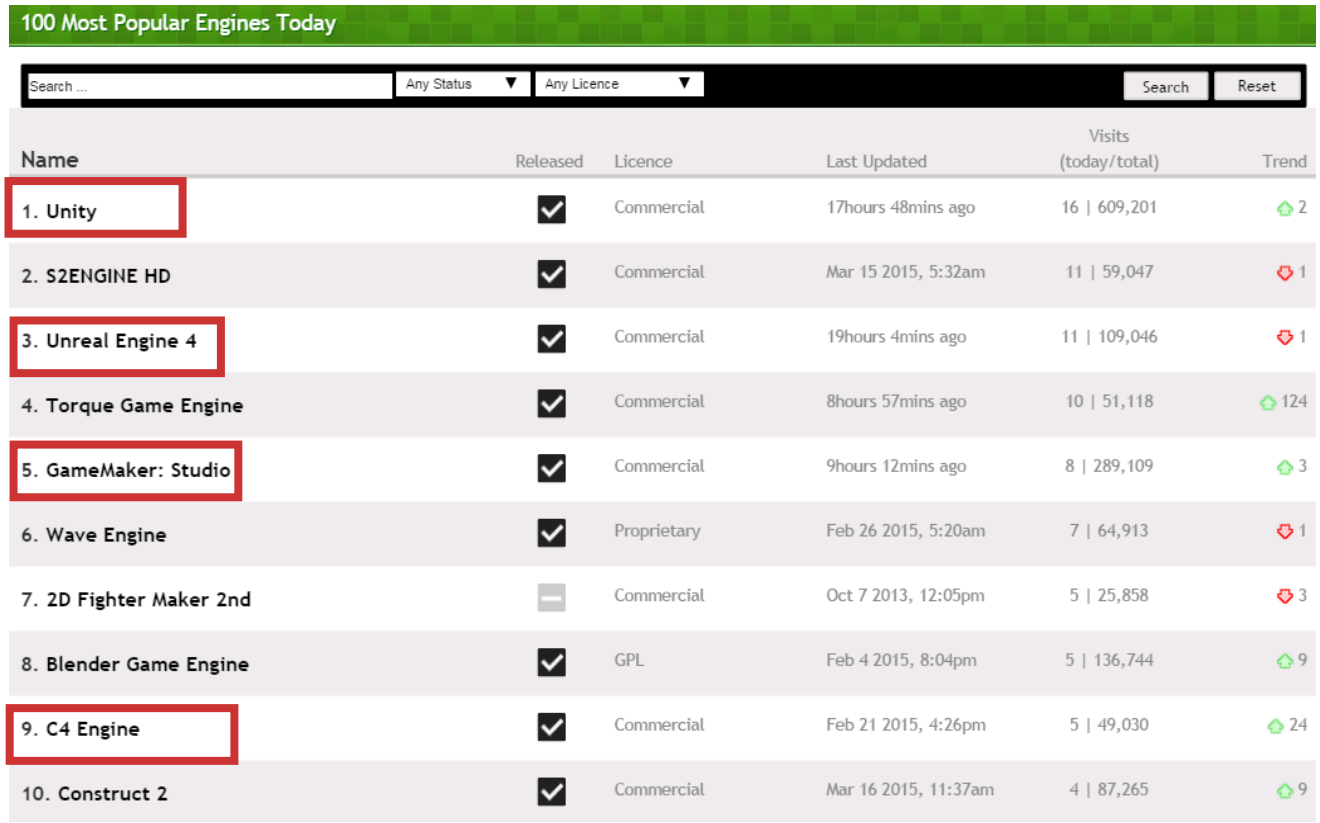
Game Engines vs. SDKs und APIs

- Game Engines benötigen oft eine enge Anbindung an die Hardware
  - Realisierung durch Plattform-unabhängige Extraschichten oder betriebssystemeigene APIs
- Kaum ein Game-Engine-Entwickler erfindet das Rad komplett neu. Meist Rückgriff auf bekannte und etablierte SDKs und APIs:
  - Datenstrukturen und Algorithmen: STL, Boost
  - Graphikhardware: DirectX, OpenGL
  - Physik: Havok, PhysX, ODE
  - Sound: Fmod, Irrklang

One Game Engine to Rule Them All?

“It’s a Jungle Out There”

- Fast mehr Game-Engines als Spiele
- 375 Engines auf DevMaster.net
- 444 Engines auf IndieDB.com



100 Most Popular Engines Today

Search ... | Any Status | Any Licence | Search | Reset

Name	Released	Licence	Last Updated	Visits (today/total)	Trend
1. Unity	✓	Commercial	17hours 48mins ago	16 | 609,201	2
2. S2ENGINE HD	✓	Commercial	Mar 15 2015, 5:32am	11 | 59,047	1
3. Unreal Engine 4	✓	Commercial	19hours 4mins ago	11 | 109,046	1
4. Torque Game Engine	✓	Commercial	8hours 57mins ago	10 | 51,118	124
5. GameMaker: Studio	✓	Commercial	9hours 12mins ago	8 | 289,109	3
6. Wave Engine	✓	Proprietary	Feb 26 2015, 5:20am	7 | 64,913	1
7. 2D Fighter Maker 2nd	–	Commercial	Oct 7 2013, 12:05pm	5 | 25,858	3
8. Blender Game Engine	✓	GPL	Feb 4 2015, 8:04pm	5 | 136,744	9
9. C4 Engine	✓	Commercial	Feb 21 2015, 4:26pm	5 | 49,030	24
10. Construct 2	✓	Commercial	Mar 16 2015, 11:37am	4 | 87,265	9

Zahlreiche Features zur Auswahl

General Info

Graphics API
OpenGL | DirectX | Glide | Software | Other

Operating Systems
Windows | Linux | MacOS | Solaris | SunOS | HP/UX | FreeBSD | Irix | OS/2 | Amiga | DOS | Xbox | Playstation | GameCube | GBA | PSP | N-Gage | BeOS | Xbox360 | PS2 | PS3 | Nintendo Wii | Nintendo DS

Programming Language
C/C++ | Java | C# | D | Delphi | Pascal | BASIC | Ada | Fortran | Lisp | Perl | Python | Visual Basic 6 | VB.NET

Status
Alpha | Beta | Productive/Stable | Inactive

Misc
Documentation

General Features
Object-Oriented Design | Plug-in Architecture | Save/Load System | Other

Game Features

Networking System
Client-Server | Peer-to-Peer | Master Server

Tools & Editors
Scripting | Built-in Editors

Sound & Video
2D Sound | 3D Sound | Streaming Sound

Physics
Basic Physics | Collision Detection | Rigid Body | Vehicle Physics

Artificial Intelligence
Pathfinding | Decision Making | Finite State Machines | Scripted | Neural Networks

Graphics Features

Lighting
Per-vertex | Per-pixel | Volumetric | Lightmapping | Radiosity | Gloss maps | Anisotropic | BRDF

Shadows
Shadow Mapping | Projected planar | Shadow Volume

Texturing
Basic | Multi-texturing | Bumpmapping | Mipmapping | Volumetric | Projected | Procedural

Shaders
Vertex | Pixel | High Level

Rendering
Fixed-function | Stereo Rendering | Raytracing | Raycasting | Deferred Shading | Render-to-Texture | Voxel | Fonts | GUI

Scene Management
General | BSP | Portals | Octrees | Occlusion Culling | PVS | LOD

Animation
Inverse Kinematics | Forward Kinematics | Keyframe Animation | Skeletal Animation | Morphing | Facial Animation | Animation Blending

Meshes
Mesh Loading | Skinning | Progressive | Tessellation | Deformation

Surfaces & Curves
Splines | Patches

Special Effects
Environment Mapping | Lens Flares | Billboarding | Particle System | Depth of Field | Motion Blur | Sky | Water | Fire | Explosion | Decals | Fog | Weather | Mirror

Terrain
Rendering | CLOD | Splatting

[DevMaster.net]

Einige wichtige Unterscheidungsmerkmale

- Kommerziell vs. Open Source vs. Kostenlos
- Unterstützte Plattformen
- Unterstützte Features/Module
- Support/Community (Online Foren, Mailing-Listen)/Dokumentation
- Unterstützte Programmiersprachen
- Extensible IDE vs. Open Class Library

Open Class Library

- Code-orientierte Entwicklung
- Meist sehr bedacht aufgebautes internes Modulsystem (Da der Entwickler damit direkt in Berührung kommt)
- Können sehr leicht modifiziert werden
- Oftmals Open Source
  - Bemerkung: Für die Unreal Engine ist der Source Code vollständig erhältlich, er darf aber nicht frei weiter verteilt werden
- Oftmal größere Hürde für Einsteiger und Gelegenheitsprogrammierer
- Beispiele:
  - Irrlicht, C4, Unreal (teilweise) 

  - Nahezu alle reinen 3D Engines wie Ogre 3D, OpenSG, Open Scene Graph

Extensible IDEs

- GUI-orientierte Entwicklung
  - Einsteigerfreundlich
  - Eher “Content”-orientiert
- Einfache Verwaltung des Contents (Texturen, Objekte,...)
  - Direkt in die Engine-eigene IDE integriert
- Eingeschränkter (bzw kontollierter) Zugriff auf die Kernfunktionen
  - Verhindert Missbrauch/Fehler
  - Verhindert aber auch eigene Erweiterungen und kann Kreativität einschränken

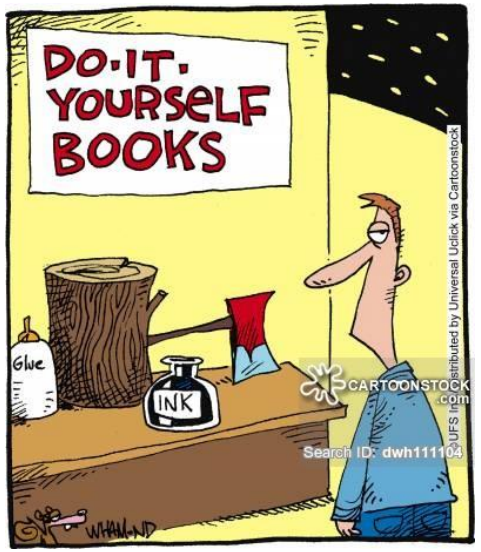
- Beispiele: Unity, Unreal

DIE “beste” Engine gibt es nicht

- Die Auswahl der Engine für ein bestimmtes Projekt ist immer sehr situationsabhängig
- Einige wichtige Auswahlkriterien
  - Plattform, Programmiersprache
  - Kosten
  - Spezielle technische Features
  - Vorerfahrung mit der Engine
  - Support
  - Bösartige Vorgaben durch den Dozenten

Vielleicht lieber alles selbst machen?

- Abhängig von Anforderungen, Resources und anderen Nebenbedingungen
  - Technische Anforderungen (z.B., will man das letzte Quäntchen Performance herausholen?)
  - Finanzielle Resourcen (z.B., gibt es genügend Startkapital?)
  - Zeitliche Nebenbedingungen (z.B., soll das Spiel in einem Monat oder erst in zwei Jahren fertig werden?)
  - Anforderungen an die Zielplattform (z.B., Browsergames mit Flash?)
  - Andere Faktoren (z.B., ist das Spiel ein Sequel und es gibt bereits Code vom Vorgänger?)
- Realität: Die meisten heute entwickelten Spiele verwenden irgendeine Art von “Engine Layer”



Gründe um komplett neu zu entwickeln

- Die technischen Anforderungen werden von keiner existierende Game Engine erfüllt (z.B. Minecraft)
- Pädagogische Gründe (weil man in erster Linie die Technologie erlernen will und erst in zweiter ein Spiel entwickeln)
- Bietet besseres Verständnis der Game Engine
  - Kann bei Bedarf einfach erweitert oder angepasst werden
- Eine eigene Engine passt genau zum Genre, in welchem man das Spiel entwickelt (kaum Performanceverluste durch Overhead, man implementiert nur Features, die man auch wirklich benötigt)
- Man will keine Lizenzgebühren zahlen
  - Eigentlich kein wirkliches Argument mehr – es gibt zahlreiche günstige oder gar kostenlose Engines → Es wird immer mehr kosten, eine eigene Engine zu entwickeln

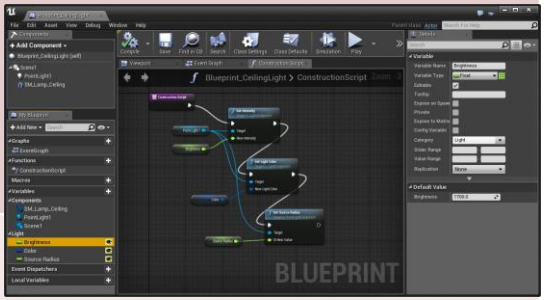
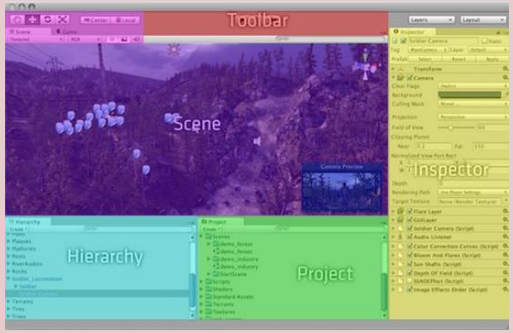
Gründe um eine Game Engine zu verwenden

- Finanzielle – Zeit/Geld reichen nicht, eine eigene Engine zu entwickeln
- Support – Bestehende Engines haben schon eine User Community und/oder eine Dokumentation
- Robustheit – Bestehende Engine haben weniger Bugs und die Codebasis ist bereits getestet
- Vorkenntnisse – Die Entwickler verfügen bereits über Erfahrungen mit einer bestehenden Game Engine

Warum Unreal Engine in dieser Vorlesung?

- Bietet sowohl GUI-orientierte Entwicklung als auch Quellcode
  - Leichterer Einstieg als beispielsweise mit C4
  - Mehr Kontrollmöglichkeiten als Unity
- Programmierung in C++
  - Nach wie vor Goldstandard im Bereich der Spieleentwicklung, schon aus Performancegründen
  - Objektorientierte Programmierung wichtiges Konzept des Software Engineering
- Direkte Integration in vollständige IDEs
  - Microsoft Visual Studio (Windows), Xcode (Mac)
- Quellcode vollständig erhältlich
  - Vereinfacht das Debugging
  - Leichte Erweiterbarkeit

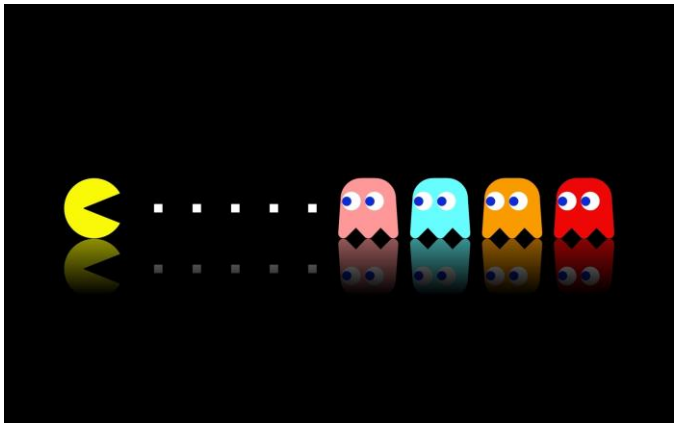
vs

	Unity	Unreal
Plattform	Windows PC, Mac OS X, Linux, Web Player, WebGL, VR(including Hololens), SteamOS, iOS, Android, Windows Phone 8, SMART TVs, as well as Xbox One & 360, PS4, Playstation Vita, and Wii U	Windows PC, Mac OS X, iOS, Android, VR, Linux, SteamOS, HTML5, Xbox One, and PS4
Code	JavaScript, C#, Boo (kein graphischer Script-Editor)	Blueprint, C++ 
Editoren		
Graphik	2D/3D	Nur 3D. Mehr Features

Abschließende Bemerkung

- Ziel der Übungen (und dieser Vorlesungsstunde) ist es nicht, EINE Game Engine (in unserem Fall der  e) bis ins letzte Detail zu durchdringen, sondern
- das Grundkonzept von Game Engines zu verstehen
  - Letztendlich sind alle Game Engines ziemlich ähnlich aufgebaut
  - Ebenso wie bei Programmiersprachen gilt: Kennt man eine, kennt man alle
- den Umgang mit diesen wichtigen Tools für die Erstellung digitaler Medien zu erlernen
- Kriterien kennen zu lernen, die Euch bei der Auswahl für die beste Engine für ein bestimmtes Projekt unterstützen

Media Engineering
Unser Projektchen



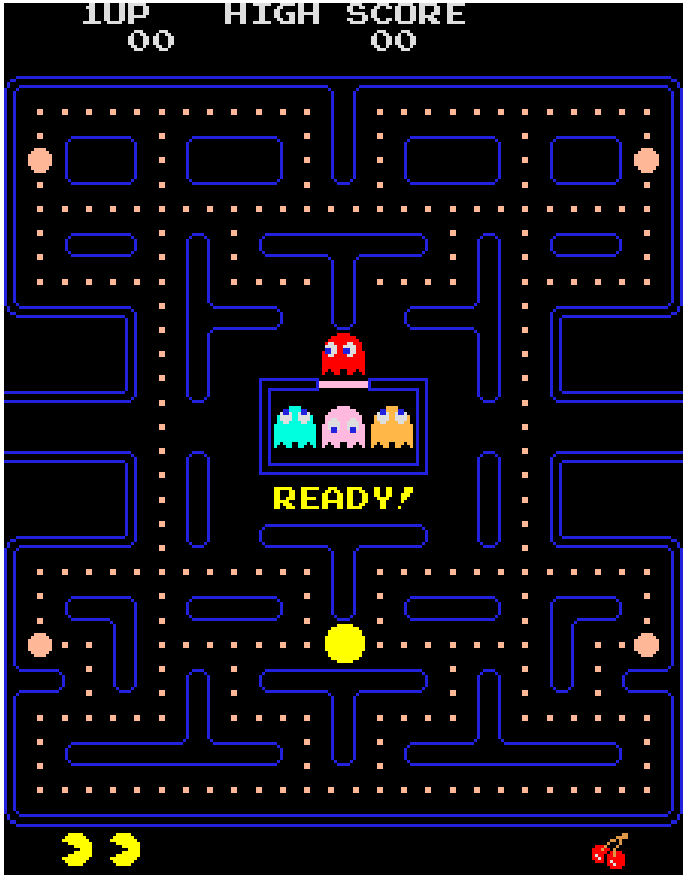
R. Weller

University of Bremen, Germany

cgvr.cs.uni-bremen.de

Ziel

- Entwicklung eines Spiels (Pacman)



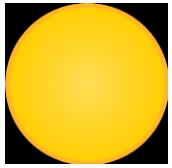
Zur Erinnerung

Was ich mir vorstelle

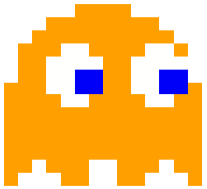
Wie Ihr es verstehen werdet

Grundprinzip

- Spielfigur

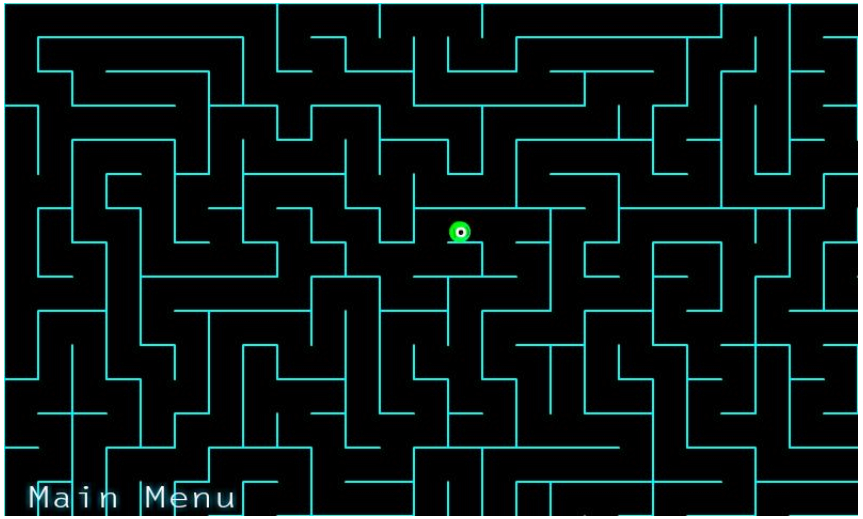


- Münzen zum Einsammeln



- Gegner

- Spielfeld

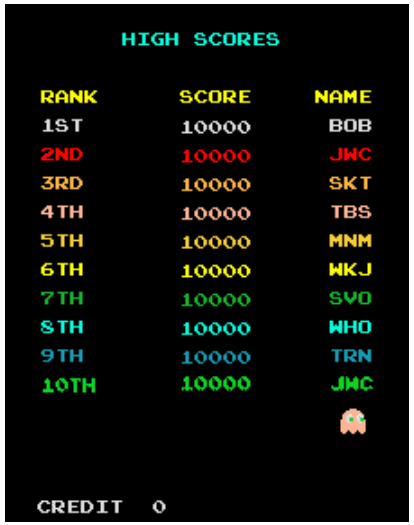


Vorgaben

- Verwendung der Unreal Engine 
- Vernwendung von C++
- Verwendung von mindestens drei eigenen C++-Klassen die nicht automatisch von Unreal generiert werden
  - Z.B. Speicherung des Spielfelds als Bitmap (2D-Array mit boolschen Werten um zu speichern, wo Spielfeld und wo Wände sind)
  - Spieler-Klasse (Speichert Ort auf der Bitmap, gefressene Münzen,...)
  - Gegner-Klasse und (ganz einfache) KI: z.B. Wenn eine Kreuzung kommt, werfe eine Münze ob rechts oder links abgebogen werden soll
- Es soll Sound verwendet werden
- GUI (Graphical User Interface): Menusystem und HUD (Heads-Up-Display)

Hinweise GUI

- Menus
  - Z.B. Optionen
  - Highscore-Liste



- HUD
  - Punktestand
  - Anzahl Leben

Viele Freiheiten

- Z.B. Perspektive
  - Top-Down
  - Third-Person
  - Ego
- Extra-Features (die keine Pflicht sind):
  - Kraftpillen
  - Kirschen
  - Tunnel
  - Animationen/Objekte
  - ...
- Zielplattform
  - PC/Mac
  - Mobilgeräte

Noch mehr Freiheiten

- Es darf auch ein anderes (einfaches) Spielprinzip gewählt werden (Tetris, Minsweeper, Sokoban, eigene Idee, ...)

- Aber: Die Vorgabenfolie ist nicht verhandelbar!
  - Unreal Engine, C++, GUI, Sound,...
- Für die eigentliche Implementationsphase sind 3 Wochen geplant! Euer Projekt muss sich in dieser Zeit realisieren lassen.

Übungen im Überblick

- Bitte nicht sofort loslaufen und rumprogrammieren
- Software Engineering bedeutet in erster Linie Texte produzieren und nicht Programmieren!
- Insgesamt 5 Übungszettel
- Kommende Woche kommt der erste Übungszettel zum Thema Anforderungsanalyse
- Weitere Übungen:
  - GUI-Design
  - UML
  - Implementation
  - Test
- Es geht nicht darum ein besonders schönes oder aufwendiges Spiel zu Programmieren! Ziel ist es am Beispiel eines Spiels das Media Engineering zu lernen!

So ein Ende wollen wir nicht...